ArcVIEW: a LabVIEW-based astronomical instrument control system

Michael C. Ashe, Marco Bonati, Steven Heathcote

Michael C. Ashe, Marco Bonati, Steven Heathcote, "ArcVIEW: a LabVIEW-based astronomical instrument control system," Proc. SPIE 4848, Advanced Telescope and Instrumentation Control Software II, (13 December 2002); doi: 10.1117/12.461433

SPIE.

Event: Astronomical Telescopes and Instrumentation, 2002, Waikoloa, Hawai'i, United States

ArcVIEW: a LabVIEW-based astronomical instrument control system

Michael Ashe[a], Marco Bonati[b], Steve Heathcote[a]

[a]SOAR Consortium, Casilla 603, La Serena, Chile
[b]Caltech, Pasedena, CA

ABSTRACT

To meet the needs of the SOAR 4.2-m telescope first-generation instrument suite, as well as new instruments for the Blanco 4-m telescope, we developed a new camera controller system called ArcVIEW. In order to provide a strong foundation and rapid development cycle, we decided to build the system using National Instrument's LabVIEW environment. The advantages of this approach centers on the tools available for rapid prototyping, integration and testing of components.

Over the past 2 years, we have taken ArcVIEW from a design document to the point of controlling two new instruments being built at CTIO. The IR imager, ISPI, will complete final testing this semester and go into use on the Blanco telescope in September 2002.
The second instrument, the SOAR Optical Imager, is due for completion this semester and will be the commissioning instrument for the SOAR telescope, for which first light is expected in early 2003.

Keywords: Instrument, Control, LabVIEW, ArcVIEW, CCD, Optical, Infrared, software, distributed

1. INTRODUCTION

The ArcVIEW or "Array Controller in LabVIEW" software suite is a modularized set of software libraries written primarily in LabVIEW that uses C for the low level driver and image data handling routines. It is described in the document: "ArcVIEW - Design Study V2.doc" [1]. It is initially intended to implement the IR and CCD array controller needs of the SOAR/CTIO instruments. It started as primarily an Array Controller, but has now evolved into a data acquisition and instrument control (DAIC) system., with components for Real Time Display, Filter Wheel control, Telescope Control System communications and other functions.

The focus of the project is to implement a generic controller architecture that uses the San Diego State University (SDSU-II) hardware controllers, commonly known as "Leach" or "SDSU-II" controllers in its initial version, but which also includes the internal and external "hooks" to implement other instruments, using other controllers, in the future, specifically multi-array, multi-controller configurations for large mosaics.

Currently implemented or planned installations of ArcVIEW include:

Instrument	Detector Type	Controller
ISPI (CTIO)	1 x 2kx2k HgCdTe	SDSU-2
Optical Imager (CTIO)	2 x 2kx4k CCD	SDSU-2
Goodman Spec (UNC)	2 x 2kx4k CCD	SDSU-2
IFU Spec (Brazil)	2 x 2kx4k CCD	SDSU-2
Spartan Imager (MSU)	4 x 2kx2k HgCdTe	E. Loh
Systems on Palomar	various	SDSU-2

In addition to the controller and communication components in ArcVIEW, there are several other software components. The following table lists other software components.

Advanced Telescope and Instrumentation Control Software II, Hilton Lewis, Editor, Proceedings of SPIE Vol. 4848 (2002) © 2002 SPIE · 0277-786X/02/$15.00

Software Component	Function(s) Overview
Optical Camera Control GUI	Implements manual setup and control of one camera. This is an ArcVIEW plug-in. The output can be sent as commands or returned for storage as part of an observation.
IR Camera Control GUI	Same, except for IR camera.
OCS Demo GUI	Enhanced demo to show capability of storing an Observation in a Queue. Manages storage, editing and running of an Observation Queue as a capability demo. Allows editing of Observations composed of the OCS demo data and Camera GUI data. Save/retrieve from file(s) and live interaction with the Queue Manager.
SOAR Communications	This implements a TCP/IP communications handler, using the SOAR libraries.
Status Panel GUI	Shows variables requested from various system components via SCL calls.
Macro/Script Handler	Processes scripts/macros written as text files. ArcVIEW will use GScript.
Image Data Manager	Implements the image data buffer and access and control of the buffer.
SDSU LabVIEW Driver	LabVIEW VIs to access the routines in the "C" driver library.
SDSU "C" Driver	Low level control of the SDSU hardware via calls to the PCI/CompactPCI interface card. An example is IRLabs DLL call library.
Telemetry	Display of any telemetry from the camera/controller combination.
TCS Communications	Uses the SOAR Communications Library (SCL) and LabVIEW code to implement message routing/response.
TCS Control Module	Specific control and response handling of TCS interaction such as would normally be associated with a full DAIC system.
"PicRead"	Reads the data out of the Image Buffer on the fly, unscrambles the data (if necessary) and routes to displays or other computers as desired.

2. SYSTEM DESIGN MODEL

ArcVIEW is a modular system consisting of a main application server that always provides TCP/IP communications services and a command processor. It loads the various modules that do the actual work at startup time (and later dynamically if commanded). These modules do the work of running the particular CCD controller being utilized, as well as control of other hardware subsystems such as filter wheels, mirrors, calibration lamps, etc. As such, ArcVIEW is extremely configurable. The current implementation at SOAR/CTIO loads a single SDSU-II controller module, but there is nothing to prevent loading two controller modules of the same type, or loading an SDSU-II and anther controller (such as Monsoon or Arcon) once an ArcVIEW controller module has been written. See Fig 1 for ArcVIEW context.

In the current implementation, ArcVIEW is written in LabVIEW 6.0.2 and 'C', hosted on PC workstations. The client(s), either GUIs or script interfaces, communicate with the ArcVIEW Main Application, and through it to the controller(s), such as the Leach SDSU-II. The various ArcVIEW software modules such as User Interfaces, Status screens, and other interfaces all route their communications through the SOAR Communication Library (SCL). The data is sent in the form of text messages that are composed of a 4-byte size header, command and optional parameter and data sections. The SCL routes packets between its various clients by setting up a series of TCP/IP – LabVIEW Queue processing "server instances" (SI). Each SI handles the communications in two directions between a sender and a recipient. When the Application starts it starts a VI to listen and spawn instances of SIs.

A Camera GUI formats the command sequences that are sent to the ArcVIEW main App. An alternative interface (or one that may be used simultaneously with a GUI) is to use the ScriptServer PlugIn to the GUI. The ScriptServer PlugIn gives direct interface to the SCL from a TCP/IP socket. This allows any of the various standard scripting languages, such as Tcl, IRAF, Perl, Python, etc to pass text commands to the ArcVIEW Application and receive the replies directly.

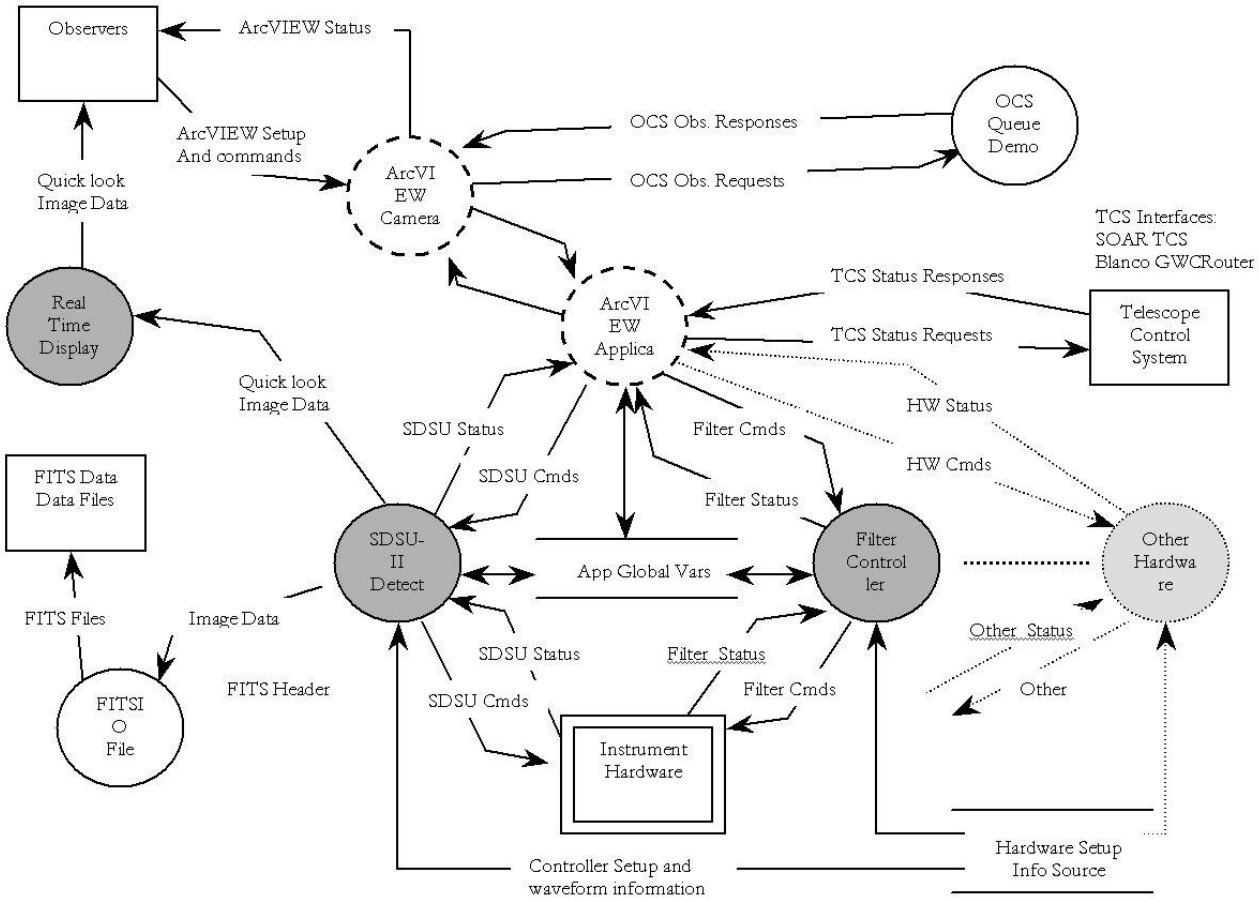
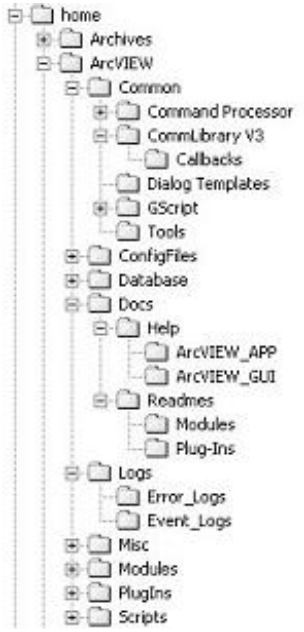


Fig 1: ArcVIEW Context and Dataflow.

3. APPLICATION ARCHITECTURE

The ArcVIEW Main Application VI consists of a fairly simple GUI and a code diagram that has five (5) main constructs: two sequence structures for startup and shutdown, and three (3) while-loops for front panel controls, incoming command processing and status retrieval and display.

The diagram in Fig 2 shows a simplified version of the Application after startup. The StartUp Sequence (SUS) on the left loads all the command processing VIs and passes a table of available commands to the main loop for use by the command processor VI. The SUS also specifies the name of the SCL communications server that the app will respond to. Above the SUS is the logging VI and the SCL Comm Server-II VI which listens for incoming connection requests and dynamically spawns Server Instances (SIs) which are copies of the CommSvrIITPLT.vi which is found in the comm library subdirectory. The SI then handles the TCP/IP communications link between the client GUI and the Command Processing loop in the center of the diagram. When messages arrive via TCP/IP from the client (i.e., the Camera GUI) the SI sends the message into the LabVIEW queue for that client link. The "Receive Command Messages Type-II.vi" on the left side of the loop waits for messages in the queue. It pulls messages out of the queue and sends them onto the Command Processor VI (CPV) (CmdParser-ProcCaller.vi) in the center. The CPV looks up the first token of the command as a lookup into the table of allowable commands and uses LabVIEW VIServer calls to run a VI of the same name as the token. The command VI being run must parse and run the command and supply an immediate return, which is returned to the Command Processor, which then passes it on to the "Send Cmd Ack Message Type-II.vi" which inserts



the reply into the return queue associated with the remote client. This queue sends the reply back to the SI which extracts it from the queue and send back to the original calling client via TCP/IP.

.

Directory Structure

ArcVIEW is installed in the "home" directory under both Linux and Windows. This maintains a common path base for both platforms, and insures commonality with the SOAR Telescope Control System.

The /ArcVIEW directory contains the libraries for the ArcVIEW Application (the server), the ArcVIEW GUI (camera GUI), the CCD Editor and the OCS Demo.

The /ArcVIEW/Common subdirectory is a central location for libraries and subdirectories of libraries that supply either a major layer (such as the SOAR Communications Library layer) or "glue" code and subroutines that are reusable over several places.

The /ConfigFiles, /Database, /Logs, and /Scripts subdirectories are each central locations for those types of items.

The /docs subdirectory has two branches for /Readme files and for HTML help. The /Modules and /PlugIns subdirectories are where Application and GUI extensions for ArcVIEW are placed.

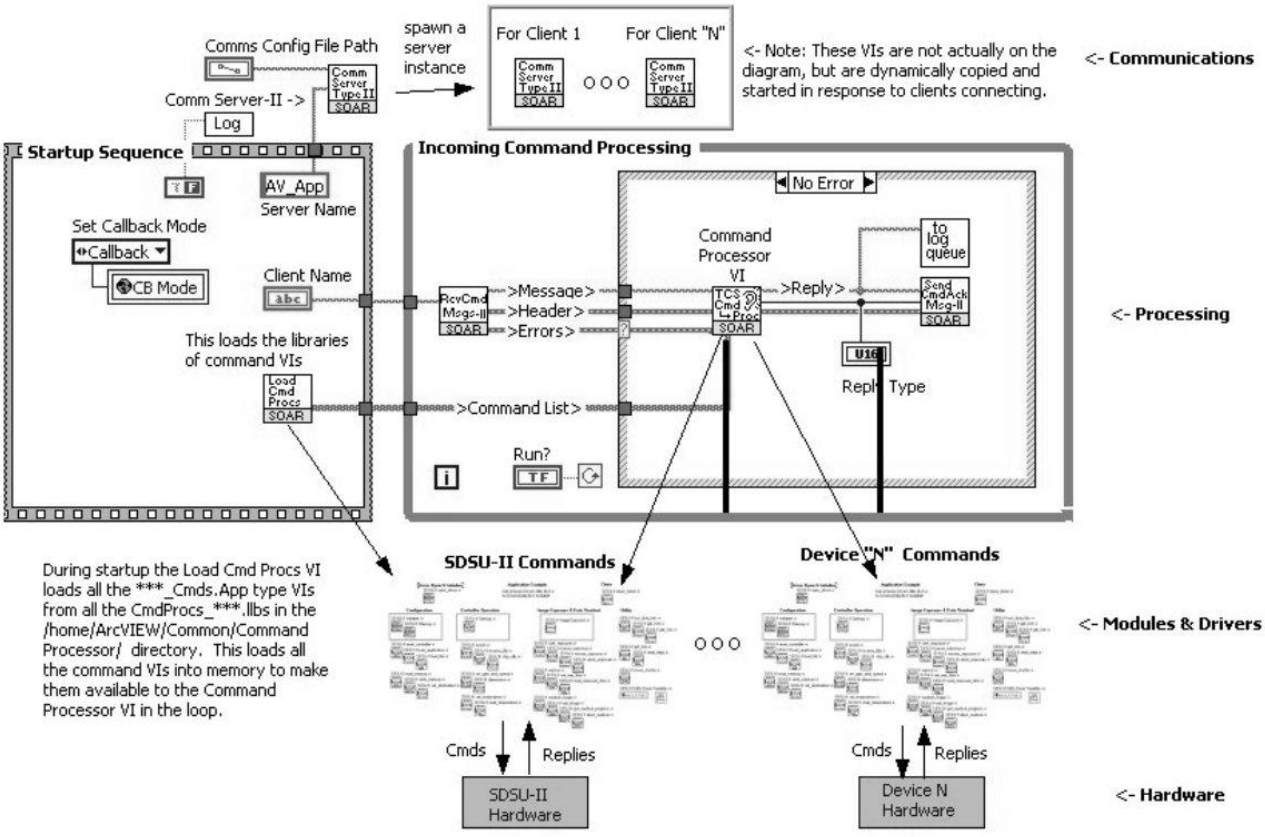


Fig 2: **Simplified ArcVIEW Application software Diagram**. Initialization sequence on left followed by main application command processing loop on right.

Strictly speaking, ArcVIEW consists of the main Application and the Data Handling processes. The Camera GUI/Client is simply one example of how to interface to the Main Application with a GUI as opposed to a scripting interface. The

ArcVIEW system consists of several levels of standard APIs, implemented in LabVIEW, for plugging in components unique to a particular telescope's instrument load. (The first controller implemented was the SDSU-II Leach controller.). A key feature of ArcVIEW is that it allows the addition of new features with little or no recoding and recompiling of the main application or any applicable Camera GUI in most cases. It accomplishes this using Modules in the Main Application and Plug-Ins in the Clients/Camera GUIs

4. PLUGIN - MODULE IMPLEMENTATION

ArcVIEW defines two types of dynamically usable code: PlugIns and Modules. Either item can be used in any ArcVIEW client or Server, but normally the PlugIns are associated with GUIs and the Modules are Associated with Instrument/Controller Applications.

Modules:	PlugIns:
1.Major Components	1.Minor Components
2.Managers, Servers, etc	2.Clients, Wizards, Tools
–Ex: Instrument CCD Controller	–Ex: Star Catalog Access
–Ex: Filter Wheel Motion Control	–Ex: Data Simulator
–Ex: Real-Time-Display	–Ex: New Report Formatter
3.Loaded mainly by Applications	3.Loaded mainly by GUIs
4.Tighter coupling	4.Looser coupling
5.Provides required functionalities	5.Provides convenience, power, integration, elegance

GUI Plug-Ins

GUI Plug-Ins add functionality to a GUI. They are dynamically loaded and can be started/opened and closed at will. Multiple Plug-Ins can be run at the same time. Plug-Ins are normally located in the directory: /home/ArcVIEW/PlugIns
The normal form of a plugin is any VI with a PI_ prefix:
PI_XXXX : all the VIs which starts with PI_ are the actual Plug-Ins. These are normally the main VI of a sub-application which is stored in subdirectories: there is one subdirectory for each Plug-In. This contains the Plug-In library, with all the VIs that are called by the main VI (which is the one which starts with the PI_ prefix. If the PlugIn does not need any extra VIs it may not need a subdirectory. In every subdirectory you will find a README file with some specific explanation

Module Overview

The Modules are components that are added to the ArcVIEW Application Server and are the components which perform the actions. This is because the ArcVIEW server Application shell does not know about detectors, TCS, filters, or other functions. It just provides communications, logging, module management and other generic services and passes the instrument commands to the modules. It is the modules that perform the actions and give the responses. The Application shell routes the responses back to the client.

The concept of "Module" is very general, in the sense that it applies to any specific thing to do, not just detectors or even specific hardware, but could be an interface module for external software to which you want to connect. A clear example of this is the GWCRouter Module, which was done for connecting the ArcVIEW-World (LabVIEW) with the GWCRouter World. This Module acts as an interface between these two worlds, and these two ways of communicating (GWCRouter and soar communication library). Analog modules could be built for communicating with ANY external environment

Every Module directory may have:
- VI libraries: have all the VIs which actually perform the actions
- C libraries: if there are some driver or specific hardware to drive, it should be here.
- README file for the explanation

- Common Global Variables – located in the /Common subdirectory of the /Modules directory. In this place should be added any variable that needs to be shared between modules. This is done in order to keep the modules independent so, if you remove one, the other ones are not broken.

Every module is "linked" to some command handler VI which is called and run by the ArcVIEW Server. For example: The AstroSDSU-II Module it is actually used through the SDSUII.vi command handler. This SDSUII.vi VI calls, using different commands, to the VIs on the Astro Module directory. Modules are located in subdirectories of the /home/ArcVIEW/Modules/ directory. Each module must have its own directory Module Management by the Application When the ArcVIEW server starts, it loads into memory all the VIs which are in the AVAPP_Cmds.App VI, so your new VI will be in memory, ready to be run. When a command arrives to the ArcVIEW Application server, it will parse the command and will try to run a VI with a name which matches **the first word** of the arrived command, and will pass to it the remaining arguments. In our example:

If a command arrives called "MONSOON SET ExposureTime 12000", it will look into its table (AVAPP_Cmds.Tbl) for a command called MONSOON. If it is there it will run the VI called MONSOON.vi, and will pass the rest of the arguments to it, SET ExposureTime 12000. This means that inside the MONSOON.vi you will need to add the necessary code for handling ALL the commands that will arrive with the MONSOON prefix. This handling will, certainly, means that you will call the MONSOON specific VIs, in your MONSOON module directory.

5. SCRIPTING CAPABILITIES

ArcVIEW employs a SriptServer VI to pass commands/responses between itself and scriptable languages and tools such as IRAF and Tcl.

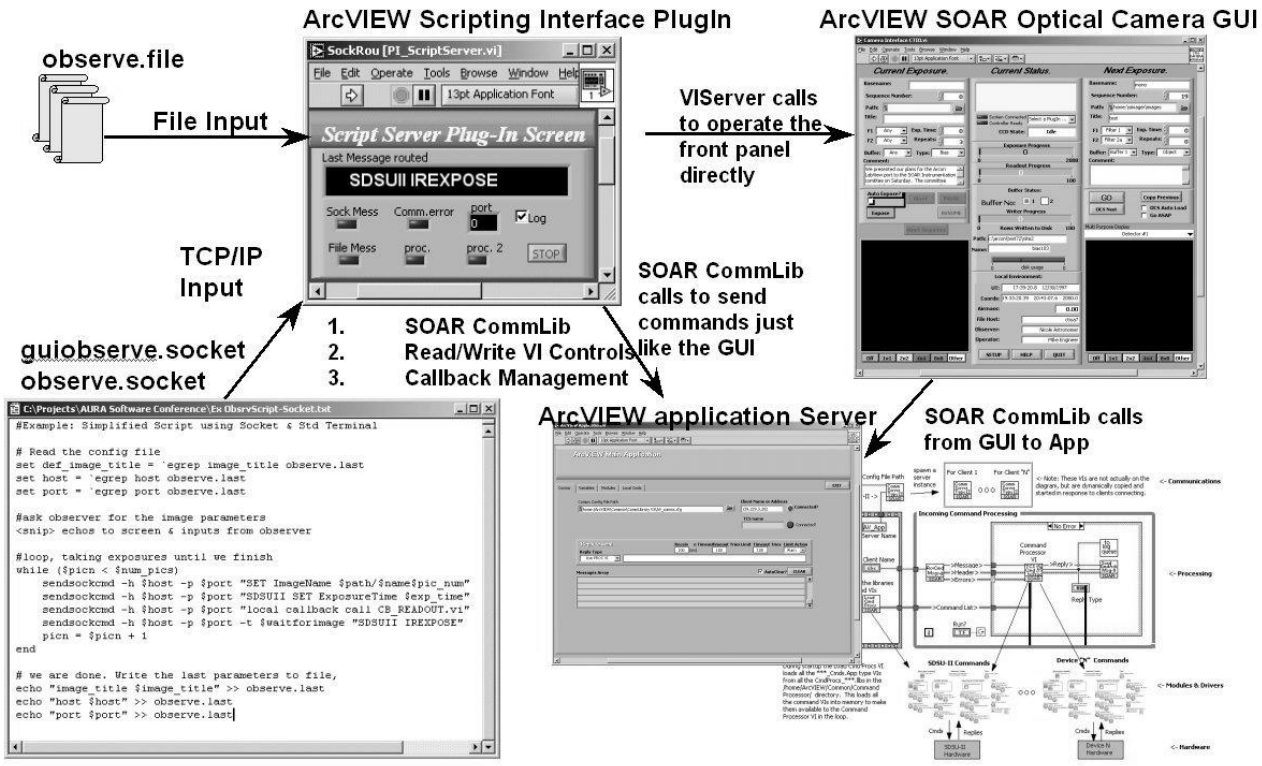


Fig 3: ArcVIEW Scripting Components

There are three "types" of scripting in the ArcVIEW

Panel Controls Record/Playback
This method allows the operator to input actions on a GUI, Every command/action that would normally get sent to the Application or Subsystem is also recorded in an array of text commands. After recording is turned off, the "Macro", or script is saved by name. Later, the GUI/Operator may recall and "replay" the script file.

TCP/IP – File ScriptServer PlugIn
This method uses a LabVIEW TCP/IP interface VI to wait on a socket, or the appearance of a file, to receive input. The ScriptServer accepts "local" commands that are directed to it's own configuration, and regular commands that either send the command to a SOAR Communications channel, or Read/Write a VI interface control directly. This is the equivalent of a "player piano" for the target GUI or LabVIEW global variable.

GScript (LabVIEW based scripting engine)
This method uses a scripting engine written entirely in LabVIEW that implements a syntax similar to structured BASIC that allows subscripts, direct LabVIEW queue read/writes and plugin extensions.

6. TCP/IP COMMUNICATIONS

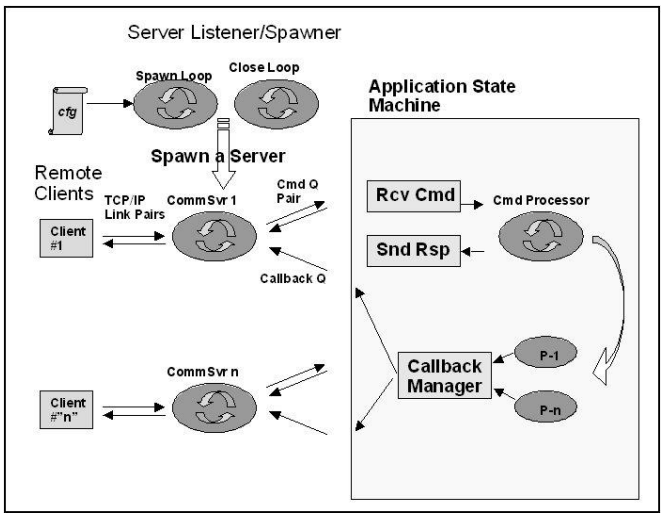
ArcVIEW uses the SOAR Communications Library for TCP/IP communications. The SOAR Communications Library (SCL) provides a native LabVIEW (platform independent) set of VIs for implementing TCP/IP connections between multiple client modules and server modules in a dynamically instantiated, non-blocking mode.

Many LabVIEW based TCP/IP server implementations either use a link-array/FOR-loop method for servicing each link sequentially or multiple copies of a server VI running on a master VI diagram.

- The former method can handle an unlimited number of links, but cannot service links in parallel, therefore it blocks other links while servicing the current link. It is also limited in how many links it can effectively handle in one second.
- The later method is non-blocking, but is limited to one TCP/IP link for each server VI on the master diagram.

The SCL solves both issues by spawning a GOOP-based dynamically instantiated server for each client machine (IP address) that initiates a connection.

Fig 4: TCS Operator Main GUI



Service handlers in the main application are connected to the SCL Server Instances (SIs) by pairs of LabVIEW queues.

The SCL has two major "modes of operation in the Type II link. These are the Callback and Inbox Modes. These modes are only applicable to the Type II. The inbox mode will normally use an additional software module called a packet router. The Callback Mode uses Callback Managers on both the client and server sides.

The main app receives incoming commands, using one or more Receive Loops and sends Immediate Responses to the calling client. If the requested action names a callback VI (i.e., the command will take time), the Command Processor passes the action off to a processing VI "P-n" and registers a callback with the callback manager.

The P-n VI signals the CB Manager when the action is complete. The CB Manager sends a callback message to the original client.

A Message Sequence Chart (Figure 5) shows the communications/command sequence of initializing the SDSU Module in the SOAR Optical Imager implementation of ArcVIEW.

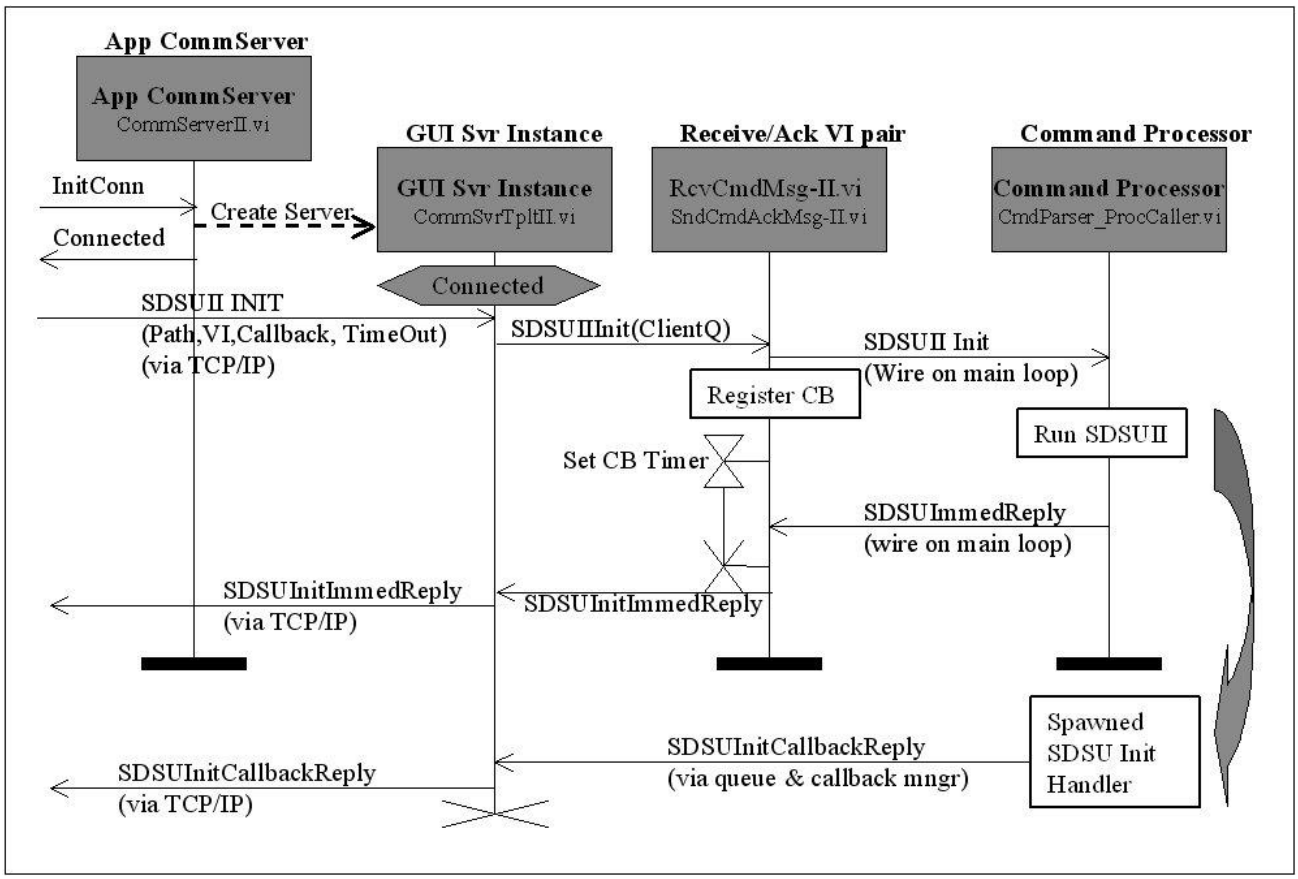


Fig 5: **Message Sequence Chart of ArcVIEW Application** software showing startup of a server instance (SI) to support the Camera Client and servicing of the first command.. to initialize and load the SDSU-II module using a VI.

7. ARCVIEW CLIENT GUIS

ArcVIEW uses LabVIEW's extensive graphical widgets along with many customized controls to implement a main GUI and the various PlugIns. The ArcVIEW Camera GUI (engineering version) was modeled after the BTC Camera GUI with modifications for SOAR/CTIO. This GUI and associated PlugIns comprised the bulk of the user interface for ArcVIEW. Recently, new GUIs were implemented for the SOAR Optical Imager, the Infrared Side Port Imager (ISPI) (for the 4.0m Blanco Telescope) and a refit on one of the Palomar Instruments. Note that it is also possible to run the Application using direct commands from it's front panel and also from a shell/scripting language using the ScriptServer PlugIn

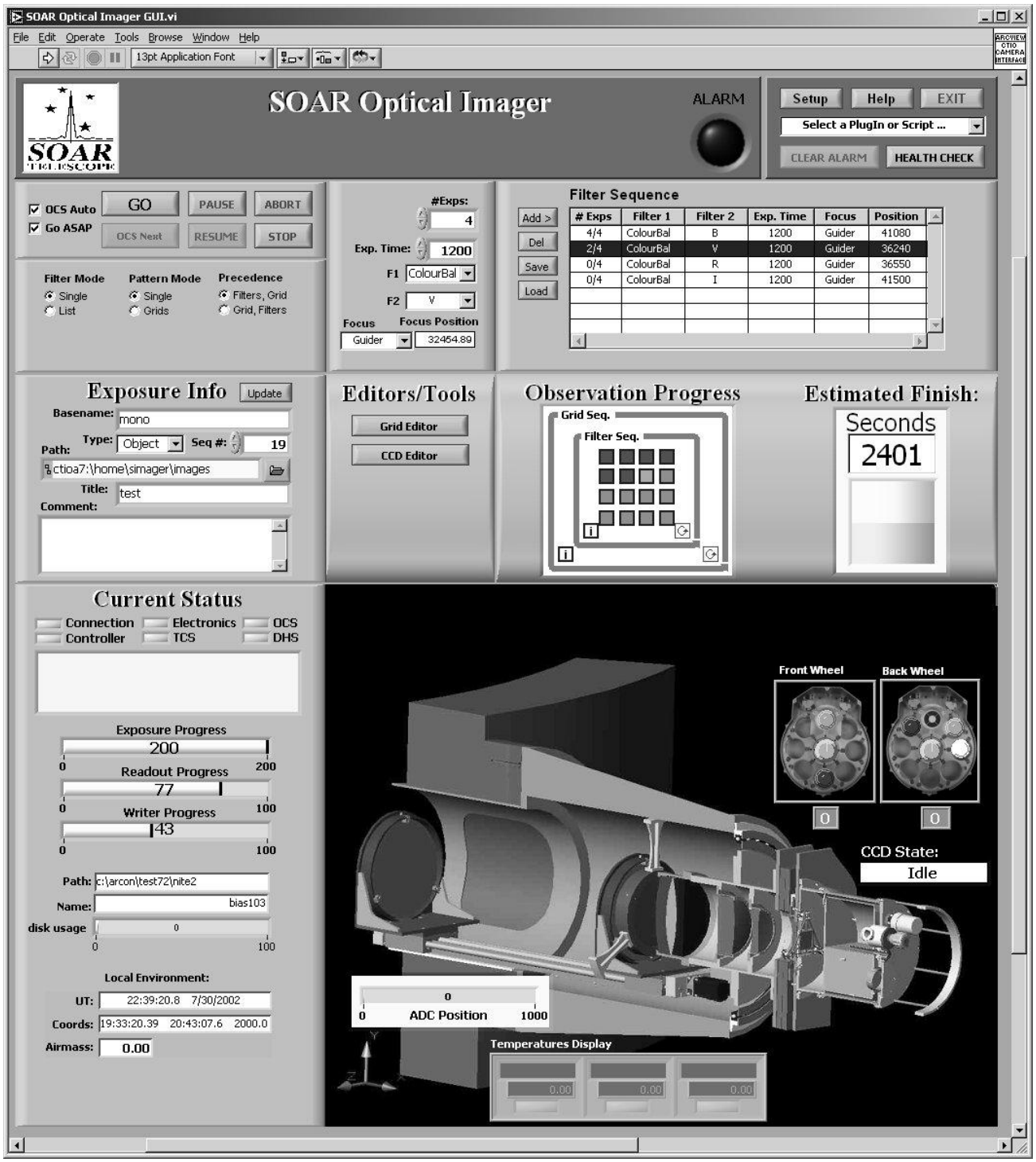


Fig 6: SOAR Optical Imager GUI

8. ARCVIEW DOCUMENTATION

Current ArcVIEW documentation consists of a 70 page Architecture/Design document, frame-based HTML Help files and text Readme files. The next phase of ArcVIEW revision will start on the Operators Manuals, both for generic ArcVIEW User Interface items, and specific addendums for the SOAR Optical Imager and ISPI user interfaces.

HTML Help for ArcVIEW

Figure 7 shows the HTML help for the ArcVIEW Application.
On the left frame of the HTML help window is a list of the top level VIs and all subVIs in an indented hierarchy. In the far left column is a letter designating what type of VI is there. The symbols are:

- "G" indicates a global variable.
- "S" indicates a standard VI
- "C" indicates a control
- "P" indicates a polymorphic VI

The left frame contains links that bring up the VI in the right two frames. The left frame also shows the path location of the VIs. The topmost frame shows all callers and callees, letting you see what VIs make use of, or are called by a particular global. The lower right frame shows the VI front panel and controls with data types and the VI description. This HTML help will be updated with time.

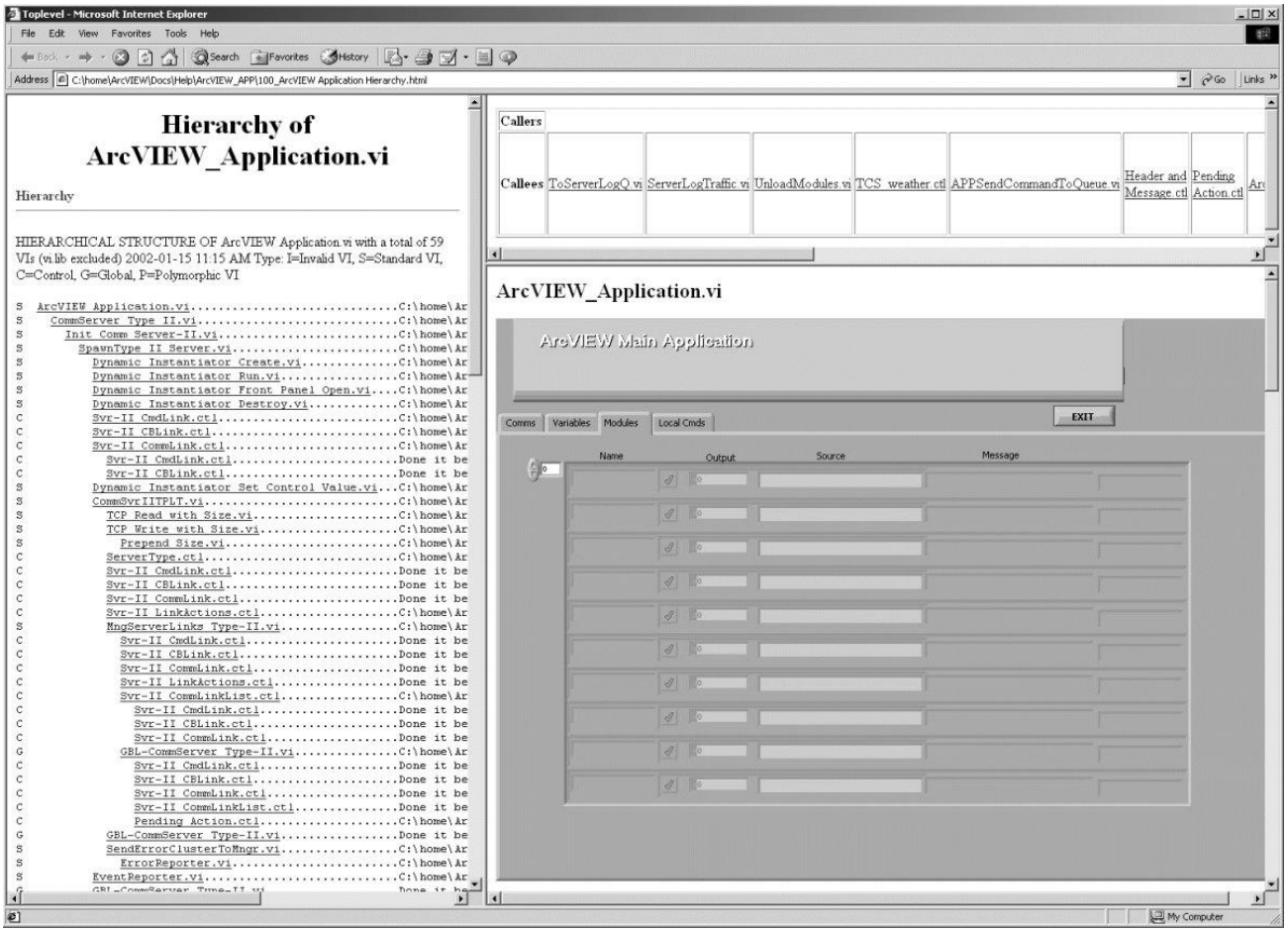


Fig 7: ArcVIEW HTML help for programmers.

9. CURRENT STATUS & CONCLUSIONS

Presently, (July/August 2002), various instruments using ArcVIEW are revising their initial GUIs. The underlying SOAR Communications Library was just revised to use a different instantiation method and improve error handling and recovery. Many new PlugIns are being added to implement new features and make the instruments easier to use. During the September 2002 to March 2003 months we expect to complete the detailed GUIs and install both the ISPI and SOI instruments. We will then integrate Remote Observing Tools[5] and prepare for using ArcVIEW and the SOI for First Light on the 4.2m SOAR Telescope.

Choosing LabVIEW as the language and using a collaborative development process resulted in a comprehensive, yet flexible instrument control system that will meet all requirements for SOAR instruments at First Light and beyond. In addition, ArcVIEW has already been ported to two other telescopes for both new and refit instrument needs. ArcVIEW is adaptable to new multi-controller large mosaics and other instrument needs and will continue to evolve for future astronomy community requirements.

10. REFERENCES

1. M. Ashe, "ArcVIEW - Design Study V2.doc" 1999-07-16, Imaginatics.
2. M. Ashe, G. Schumacher, "The SOAR Telescope Control System: A Rapid Prototype and Development in LabVIEW", Advanced Telescope and Instrumentation Control Software, pp. 48-60, Hilton Lewis, SPIE, Washington, 2000.
3. G. Schumacher, et al, "SOAR TCS: From Prototype to Implementation", SPIE 4848-24, this proceedings.
4. M. Ashe, et al, "SOAR Control Systems Operation: OCS & TCS", SPIE 4848-30, this proceedings.
5. G. Cecil, et al, "Remote Use of the SOAR 4.25m Telescope with LabVIEW", SPIE 4845-12, this proceedings.
6. R. Probst, et al, "ISPI: the Infrared Side Port Imager for the CTIO 4-m telescope.", SPIE 4848-58, this proceedings.
7. A. Walker, et al, "The SOAR Optical Imager", SPIE 4841-31, this proceedings.